REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*
31-10-2004

2. REPORT TYPE
SBIR Phase I Final Report

3. DATES COVERED *(From - To)*
03 May 2004 - 31 Oct 2004

4. TITLE AND SUBTITLE
Recovery Algorithms for Ship Survivability

SBIR N04-134 Phase I Final Report

5a. CONTRACT NUMBER
N00014-04-M-0112

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Broadwater, Robert, P, Ph.D.

Russell, Kevin, J

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Electrical Distribution Design, Inc.
311 Cherokee Drive
Blacksburg, VA 24060

8. PERFORMING ORGANIZATION REPORT NUMBER
ONR SBIR N04-134 Phase I

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Ballston Tower One
800 North Quincy Street
Arlington, VA 22217-5660
Attn: Katherine Drew, ONR 334

10. SPONSOR/MONITOR'S ACRONYM(S)
ONR 334

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; SBIR Report, Distribution Unlimited

20041108 168

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Phase I work performed by Electrical Distribution Design, Inc (EDD) evaluated use of a Linear Graph Iterator based approach for reconfiguration and recovery modeling and analysis of shipboard systems. This approach was originally developed by Virginia Tech and EDD for the Power Utility Industry. Several simplified ship system arrangements for electrical distribution, firemain and chill water were modeled and analyzed. Included in the models were multiple reconfiguration paths, prioritized loads, and relatively short loops and low cable impedances. In addition to demonstration modeling, EDD completed a significant amount of recovery analysis problem definition and integrated systems analysis software architecture definition work. Results demonstrated that: (1) EDD's Linear Graph Iterator model based approach is well suited for structuring shipboard system recovery analysis for both design and control; and (2) Significant synergy potential exists for using a Linear Graph Iterator based approach to structure concurrent Utility, Navy and Homeland Security critical infrastructure system reconfiguration and recovery design and control research.

15. SUBJECT TERMS
Recoverability, Reconfiguration, Recovery, Model Based Reasoning, Automated Survivability, Automated Damage Control, Linear Graph Iterator

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Kevin Russell
U	U	U	UU	30	19b. TELEPHONE NUMBER *(Include area code)* 540-951-6942

Recovery Algorithms for Ship Survivability

SBIR N04-134 Phase I Final Report

Contract No. N00014-M-0112

(03 May 2004 to 31 Oct 2004)

Navy Small Business Innovation Research Program

Electrical Distribution Design, Inc.
311 Cherokee Dr
Blacksburg, VA 24062

ONR N04-134 Phase I

Office of Naval Research

Table of Contents **Page**

INTRODUCTION

The Phase I work performed by Electrical Distribution Design, Inc (EDD) evaluated use of a Linear Graph Iterator based system modeling and analysis approach for reconfiguration and recovery modeling and analysis of shipboard systems. This approach was originally developed by Virginia Tech and EDD for the Power Utility Industry. Several simplified typical ship system arrangements for electrical distribution, firemain and chill water were modeled and analyzed. Included in the analysis were multiple reconfiguration paths, prioritized loads, and relatively short loops and low cable impedances.

Results demonstrated that:

- A Linear Graph Iterator (LGI) model based approach is well suited for structuring shipboard system recovery analysis for both design and control.
- EDD's LGI based approach could be extended to provide a unified framework for combining detailed electrical, mechanical, piping, ventilation, compartmentation, monitoring and control system models together into a "system of systems" recoverability model. The resulting model provides for optimal integrated system design and real-time survivability management and control for Naval ships.
- Significant synergy exists for using an LGI based approach to structure concurrent Utility, Navy and Homeland Security complex system reconfiguration and recovery design and control research.

In addition to demonstrating potential shipboard use of LGI based recoverability analysis, Phase I work also provided foundational problem definition and concept evaluation work necessary for formulation of a viable concurrent military and commercial strategy for development of complex critical infrastructure optimal system design and real-time control that includes distributed agents, field programmable monitoring and control agents, and model based reasoning, control and monitoring.

RESULTS

Figures 1 and 2 illustrate application of the approach for a simple ship system equipment casualty involving the failure of two cable sections transiting a damaged compartment. The example system was drawn using EDD's existing Distribution Engineering Workstation (DEW) graphic user interface, which was specifically designed for power utility distribution system work. DEW's algorithms could be modified for use with existing shipboard system interfaces, or be used to support development of a new dynamic Damage Control plate interface that could be used for real-time decision support, control and training. The thick red and green lines represent firemain and chill water piping. The brown lines represent electrical distribution cables. The isometric box arrangement represents compartmentation. To perform reconfiguration for restoration the user selects the failed components, and then tells the system to perform reconfiguration analysis. The system first identifies the switches to operate to isolate the affected components. For the example, crew members would be directed to open the circuit breaker feeds for the affected cables, located at the No. 1 and No. 2 Ships Service Switchboard, and the ABT (Automatic Bus Transfer) switches at the load would automatically shift when power supply at the main switchboards is secured.

The system then identifies available alternate power feeds from the emergency switchboard and evaluates voltage, current and power constraints in order of set load priorities. For the example, the Air Search Radar load was given a higher priority than the Surface Search Radar. The constraints for the Air Radar load were met, so the system closes the ABT switch connecting the

Air Radar to the emergency switchboard. Connecting the Surface Radar would exceed the set power rating for the emergency switchboard, so the ABT alternate power supply is left open.

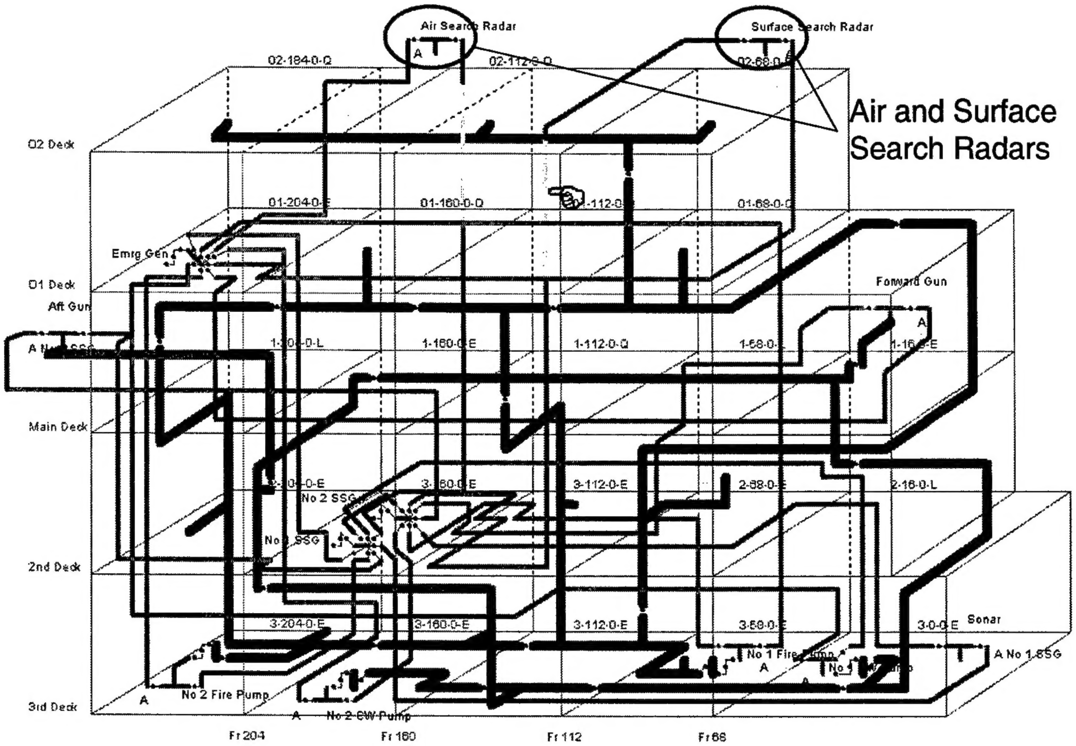


Figure 1. Operator identification of failed cable sections (highlighted in yellow).

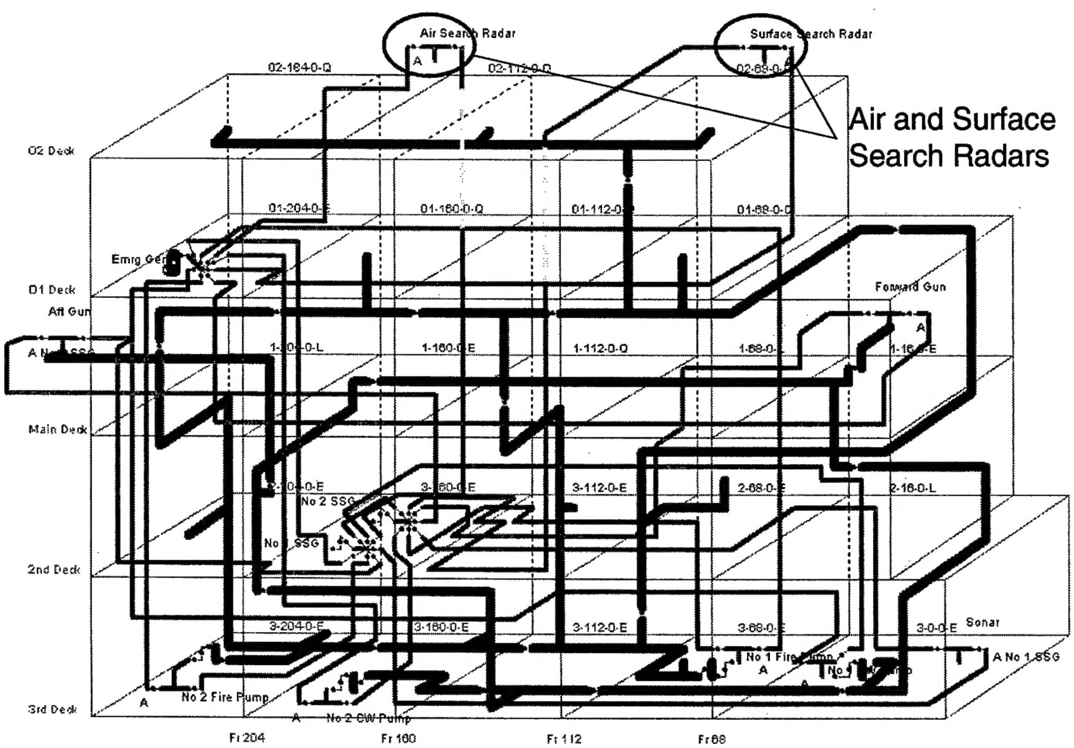


Figure 2. System identification of components affected by casualty. (Highlighted in purple).

A preliminary observation from this work is that the circuit segmentation and highlighting that DEW uses as a natural part of its analysis greatly increases the amount of usable information that can reasonably be displayed in one area. This is critical to the development of effective multi-system decision aid displays and training simulations.

Load Priority Management

For the example show in figures 1 and 2, load priorities were set using a dialogue box that assigns load priority by primary mission groups, which are associated with sets of vital system equipment which directly support those missions. See Figure 3. This concept works well for primary equipment like radars and guns, but not for the reconfigurable auxiliary system components that support them. Load priorities for reconfigurable components could be dynamically derived and assigned according to current equipment, compartment and mission priority status by combining DEW's existing linear graph iterator based trace algorithms, and the creation of a dependency component that transfers reconfiguration related information between tangential elements at the contact points between interdependent systems. Detailed architecture definition for dependency elements is provided in the following section.

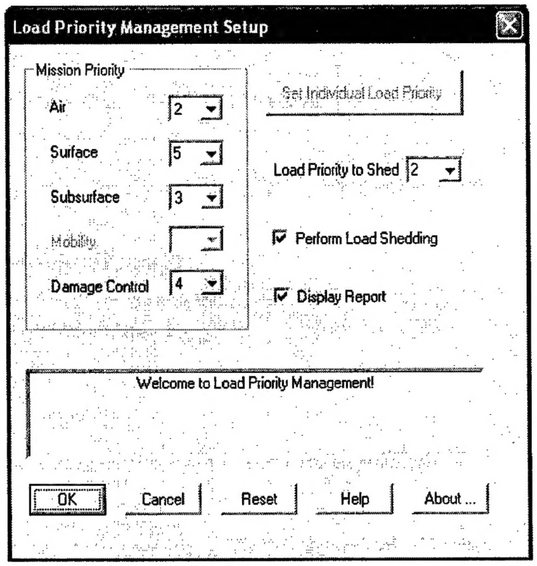


Figure 3. Load Priority Management

Figures 4 and 5 illustrate a simplified interdependent system mission priority driven reconfiguration example for compartment damage that affects multiple systems. For the example, compartment damage affects a chill water pump that is supplying cooling to a high priority weapons system. A power supply cable that runs through the damaged compartment is destroyed, which affects alternate power supply for a standby chill water pump located forward in an unaffected compartment. The key to reconfiguration for restoration for this problem is giving the surviving power feed to the standby chill water pump the same priority that was assigned to the weapons system that the aft chill water pump was providing cooling to before the casualty.

For this example, CoTree dependency elements are added to the model that associate the forward chill water pump with its electric motor, and the forward gun with its chill water cooler. The dependency elements transfer assigned priority information in the backward trace direction from the forward gun to the chill water cooler, and the chill water pump to the chill water motor. The chill water system trace algorithm would pick up this information from the chill water pipe section feeding the forward gun cooler, and then use it to generate a list of priorities for all of the components that the forward chill water pump could be reconfigured to serve, given current mission status, equipment status, and compartment status information.

The chill water pump dependency element then takes the highest priority number from the list of possible elements the pump could serve, and then assigns that priority to the electrical power supplying the chill water pump motor. The electrical distribution system reconfiguration algorithm then uses the pump motor's priority information to perform prioritized load shedding and reconfiguration for restoration according to overall total ship status and mission priority conditions.

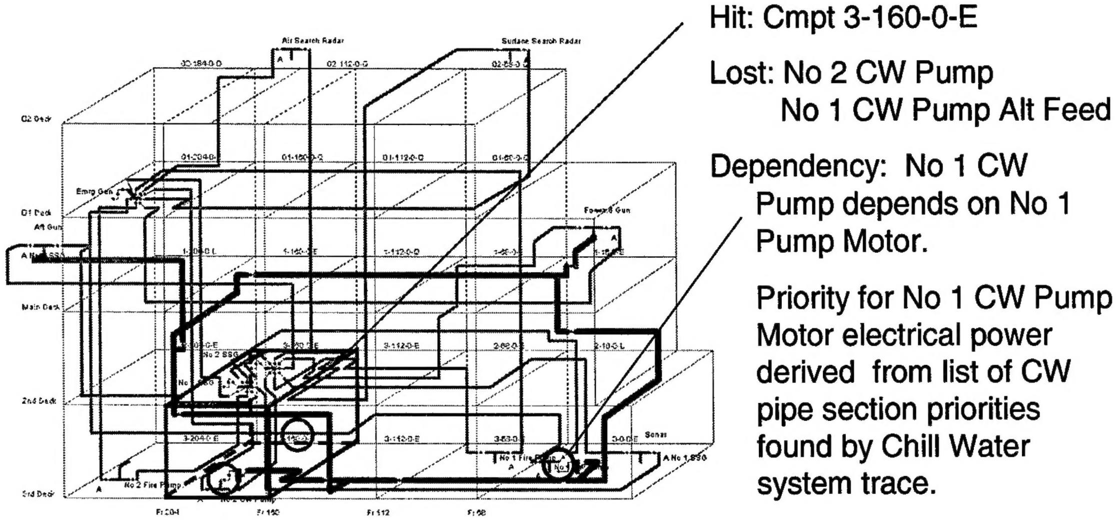


Figure 4. Multi-System casualty involving chill water and electrical distribution.

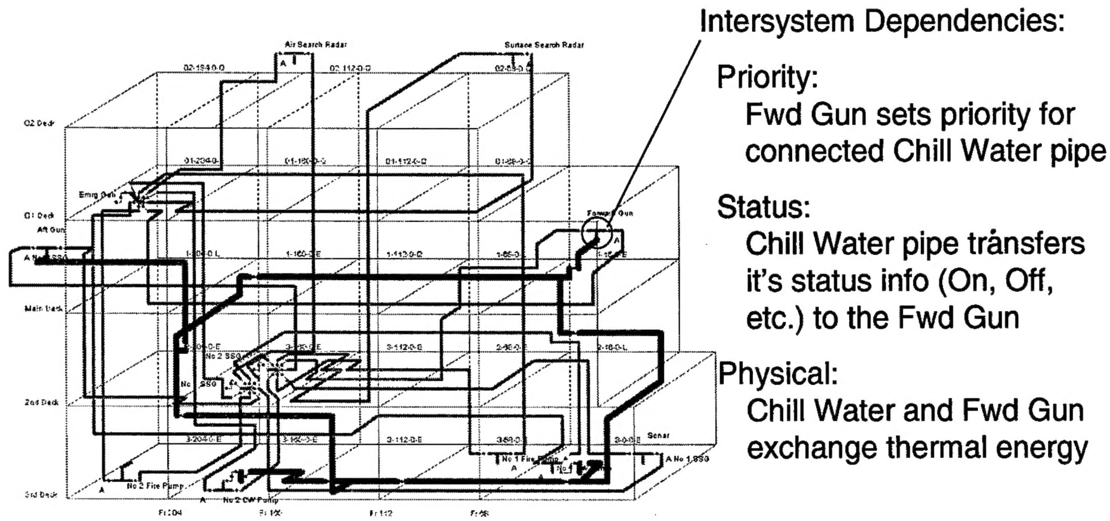


Figure 5. Modeling intersystem dependencies at the component level.

Doctrine Related Personnel Actions and Manual Equipment Operation

A significant part of evaluating intersystem reconfiguration and recovery for both design and control involves evaluating personnel actions like operating manual motor controllers, remote valves and MBT (Manual Bus Transfer) switches, and identifying routes for rigging casualty power and fire hose jumpers around isolated electrical and firemain system damage. The dependency element based architecture described above could be used to incorporate compartmentation and personnel routes as an interdependent part of the total ship interdependent systems network. See Figure 6. Across variables for these components would be distance. Through variables would be transit and operation times. (Note, in other physical systems examples of across variables include voltages, pressures, and temperatures, and examples of through variables include current, fluid flow, and heat transfer.) System equations for personnel routing and compartmentation would be similar to those used for other physical systems. Dependency elements would associate end path elements with equipment remote and local actuators and controls. This could also be applied to evacuation design and safety system

operation for high-rise buildings and large cruise ships. Doctrine related activities like battle messing and safe routes for personnel transfer could also be analyzed.

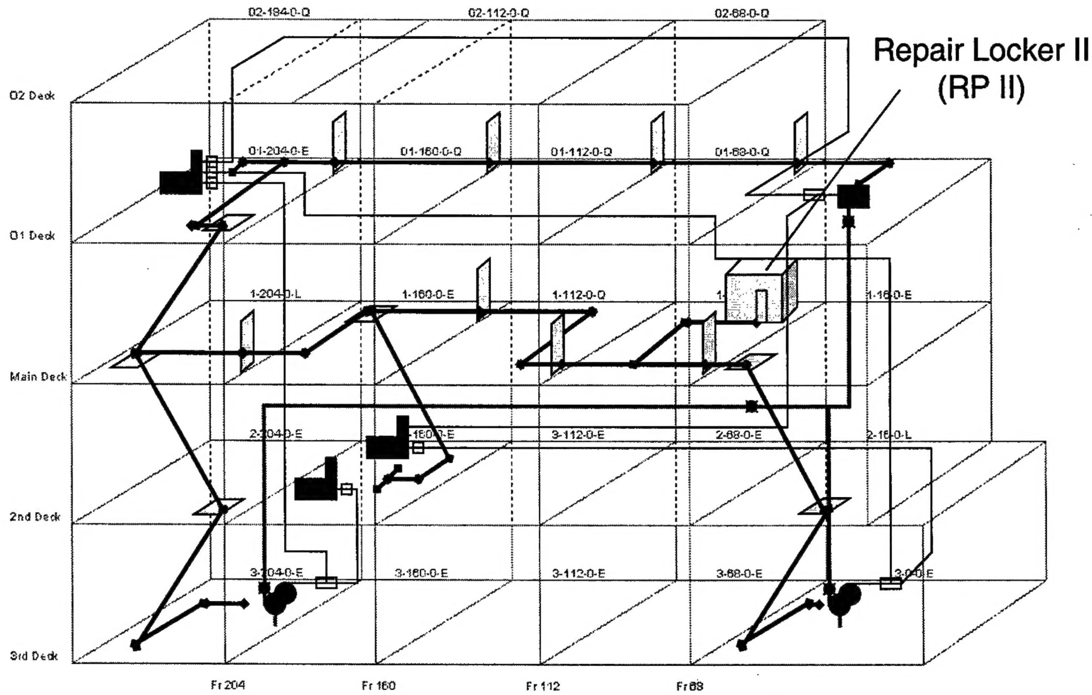


Figure 6. Modeling personnel routes, casualty rigging routes, manual equipment operation.

Interdependency Junction Component (IJC) Definition

The following section defines a generic architecture for extending EDD's current power utility analysis and control software for use with systems from multiple engineering problem domains. EDD's LGI based algorithms are built around physical modeling principles that were originally developed for interdisciplinary systems analysis in the 1960s. This means that most of the software development and analysis approach work EDD has done for the Power Utility Industry is readily transferable to other system areas like integrated shipboard system design and control.

For example, for load flow EDD's DEW software uses a system's network topology, plus a combination of on-line monitoring and time varying load history information to determine initial estimates for voltages across feeder paths along with the currents that pass through them. Using physics based component equations, DEW calculates component voltage drops, updates currents and voltages across the system, and then repeats the process until the solution converges. For fluid flow, the component algorithms are similar to load flow and the topological equations are the same. Flows at a junction must sum to zero and the pressure drops around a loop must also sum to zero. The main difference is that the component equations are highly nonlinear in comparison with electrical circuits. This nonlinearity does not affect convergence and has been shown to work in nuclear power plant modeling [1]. The same basic network approach could be used for any steady state or transient flow process that can be modeled as a linear graph of networked components.

Extending these concepts to include concurrent analysis of interdependent systems across multiple engineering problem domains would require the creation of an Interdisciplinary Junction Component (IJC) that defines and manages connection points between systems. This concept

may be viewed as an extension of the Linear Graph based arc-node network representation approach. See Figure 7.

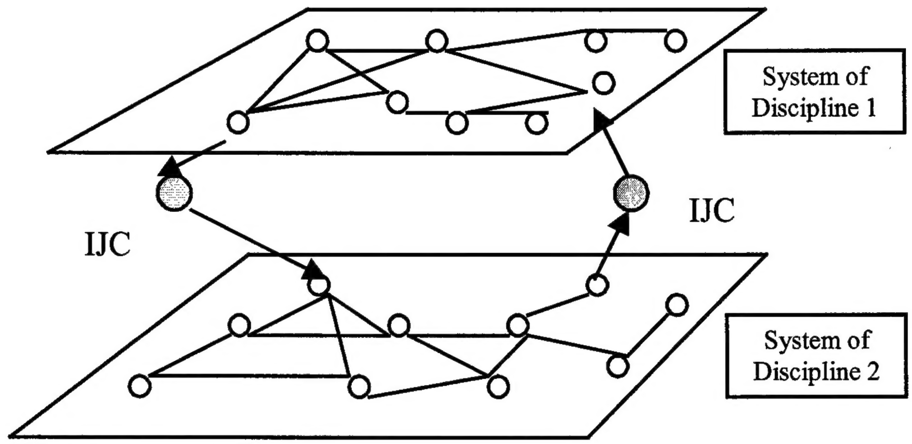


Figure 7. Interconnecting systems through IJCs

An IJC defines what needs to be modeled to link two systems together for a particular type of analysis. Using the concepts of class and inheritance, IJC objects could be formulated with progressive degrees of complexity as shown in Figure 8. Dependencies can be categorized into three types: Energy, Information (which includes monitoring and control), and Environment.

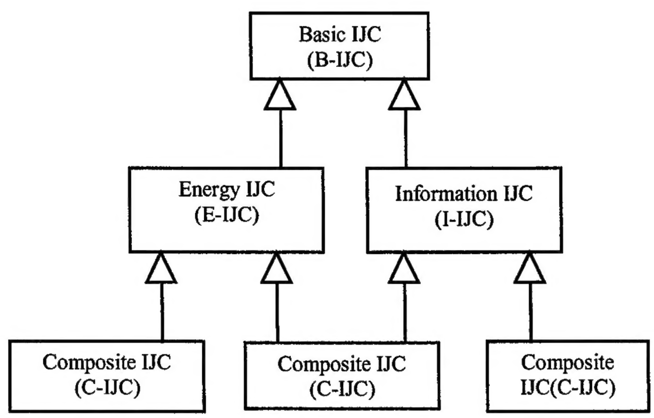


Figure 8. Inheritance relationships between different kinds of IJCs

Basic IJC: The Basic IJC (B-IJC) defines that a dependency exists, the two points between which the dependency exists, and the direction of the dependency. For example, a B-IJC would be used to propagate failure and priority information between an electrical motor and a pump.

Energy IJC: The energy IJC (E-IJC) inherits all the properties of B-IJC and in addition, defines how energy from one system is transformed into energy in another. For example, a pump is modeled as a load in an electrical system and a source in a fluid system. The E-IJC is used to transfer energy information between the electric and fluid system ends of the pump. A control or casualty action in one system will have some energy related affect on the other through the pump-motor dependency component that both systems share. A voltage drop in the electrical system will affect the pump's motor speed, which in turn will change the output of the pump. The fluid system flow analysis application uses the new pump speed to iteratively recalculate pressure and

flow across the entire fluid system. The converged result for pump flow and pressure will in turn have some impact on the electrical load at the pump motor, which in turn will trigger a new load flow analysis run on the electrical side. This analysis exchange will continue across the dependency point defined by the E-IJC until both sides converge to within some set difference tolerance.

The E-IJC shown in Figure 9 could be used to depict an electric pump, an electric fan or an electric compressor. The E-IJC interconnects the electric load model (with P, electric power, as the variable) and the pressure source model (with h_p, pump head, as the variable) together by defining the relationship between them, for example: $P = h_p * Q * \gamma$, whereas Q is the volume flow rate, and γ is the specific weight of the fluid.

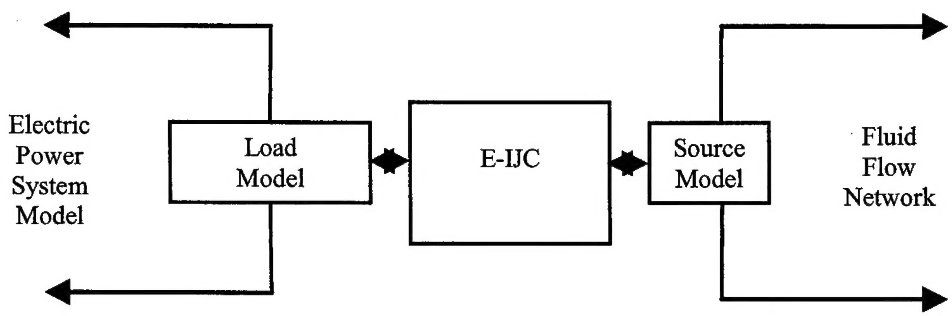


Figure 9. E-IJC for electric pump, electric fan, electric compressor

Power converters may be viewed as a combination of E-IJCs. Figure 10 shows an E-IJC combination where electric power is drawn from the AC system (through the load model) and supplied to a DC system (through the source model).

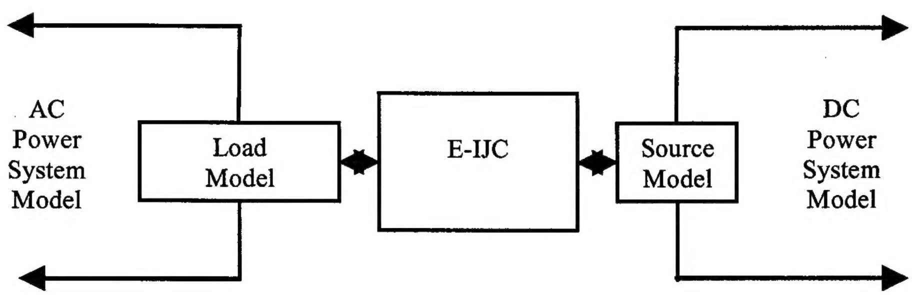


Figure 10. E-IJC for AC/DC rectifier

E-IJCs may also be used to link more than two systems together, as shown in Figure 11.

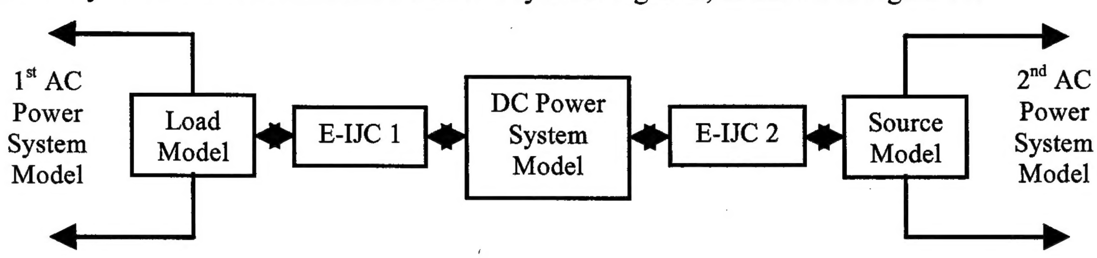


Figure 11. Three systems interconnected with E-IJCs

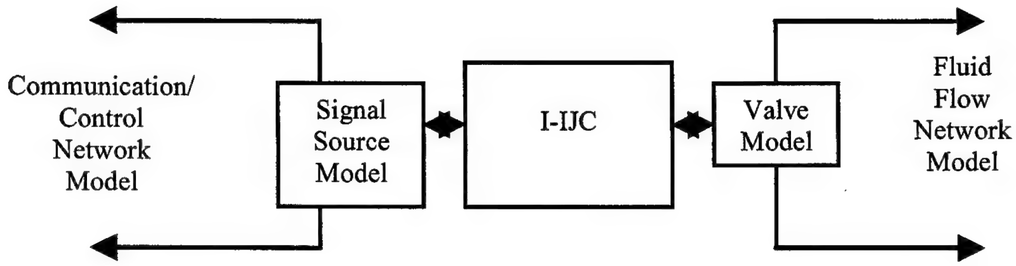


Figure 12. I-IJC for a remote-controlled valve

Information IJC: Information IJCs (I-IJC) model the dependency relationship between controlled devices and the control signals that travel through communication or control networks. Figure 12 shows how an I-IJC can be used to model the information dependency between control, a communication network, and a valve in the fluid flow network. The I-IJC inherits all the properties of the B-IJC and in addition, defines how information contained in control signals can be used by the controlled devices. If the simulation is only concerned with whether a control signal is available to the controlled device and the details of how the control message is implemented are not important, then the B-IJC can be used instead of the more complicated I-IJC.

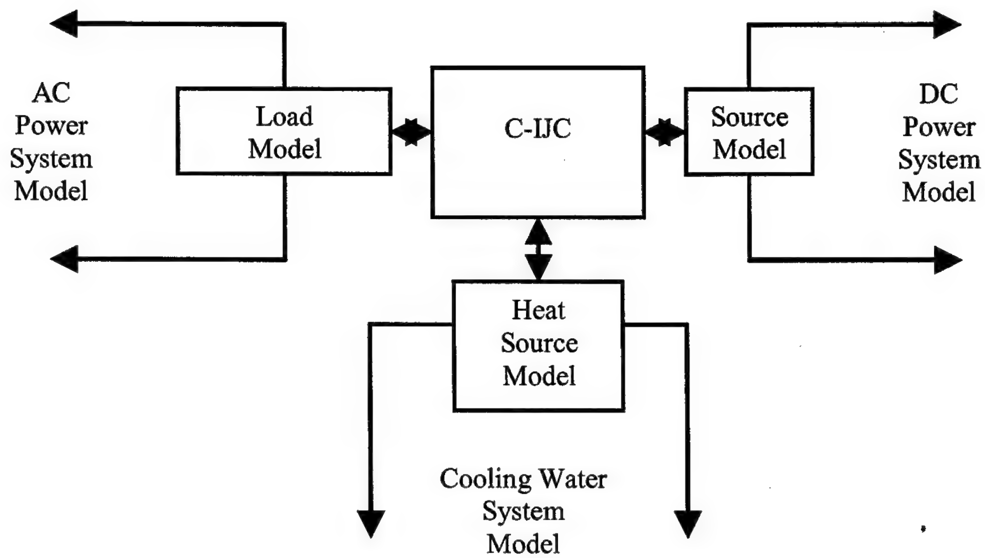


Figure 13. C-IJC : AC/DC converter with heat removal system included.

Composite IJC: A composite IJC (C-IJC) includes more than one E-IJC or I-IJC. It is used to describe interdependency relationship between three or more systems. A C-IJC inherits all the properties of the E-IJC or I-IJC and it includes a new relationship that describes the additional coupling. For a large converter, this process could be used to model the interfaces between converters and the interdependent cooling and control system components that support them. See Figure 13. The C-IJC model would make it possible to simulate a pipe rupture in some remote part of the cooling system, and then evaluate the affect that rupture has on the converters load capacity, which in turn could affect the electrical system that the converter supplies power to.

Environment Object: An environment object would be used to model environmental dependencies between components and the rooms or compartments they are located in. The same concept could be used for representing equipment that contains modeled components from other

systems. This could also be used to coordinate concurrent engineering casualty control and compartment damage control operations that affect the same systems and components. For example, electrical components located in a compartment that is on fire have a higher probability of failure, and may need to be secured or reconfigured to protect firefighting personnel working in the compartment. At the same time, keeping these same electrical cables and equipment energized may be critical to the survival of the ship because of some other threat. This same concept could also be used to associate a piping rupture with damage control stability analysis, casualty power rigging, and hose rigging. It could also be used to determine operation times that include personnel transit between locations to operate remote circuit breakers and valves.

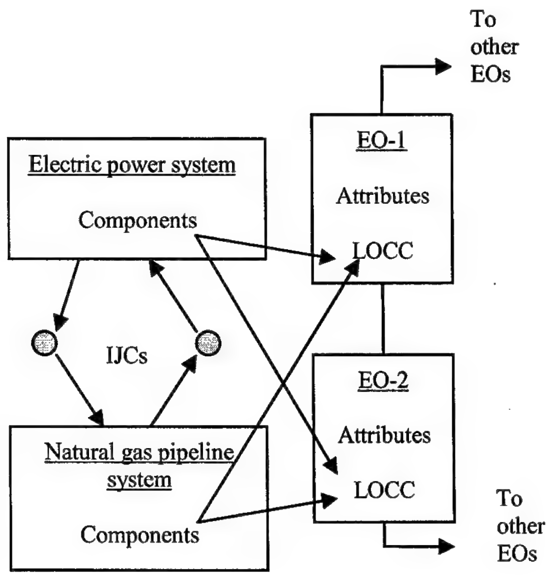


Figure 14. Environmental Object

Figure 14 shows how an EO could be applied to the modeling of electric power systems and natural gas pipeline systems that have collocation dependency. Each EO is characterized by a set of attributes that include coordination of the boundaries that define the scope of the EO; environmental parameters such as temperature; identifiers of all neighboring EOs; and a list of identifiers for all components that go through or reside within the EO (this list is referred to as LOCC, List Of Components Contained). When the environment represented by an EO is damaged, the systems that have components in the LOCC will be affected and appropriate actions, such as reconfiguration of the relevant networks, will need to be invoked.

Advanced Ship System Design

The following model was generated to provide an initial evaluation for using the approach to test and design future shipboard arrangements, such as a switched offset loop inline interrupter based electrical distribution system, which is laid out similar to a firemain system on a Frigate or a Destroyer.. This arrangement produces good tradeoff characteristics between survivability and cost. Segment protection and reconfiguration control for an electrical system of this type would be difficult, but technology advances are beginning to make it a concept worth evaluating. The system model shown below uses the same loads used for the simplified traditional shipboard system arrangement model shown in the previous section. The new system feeds the modeled electrical loads using two parallel run offset loops that include a large number of reconfigurable sections and isolation switches. EDD's DEW software was used to simulate reconfiguration for

restoration, and also to perform automated reliability to evaluate the effect that switch placement had on overall system reliability.

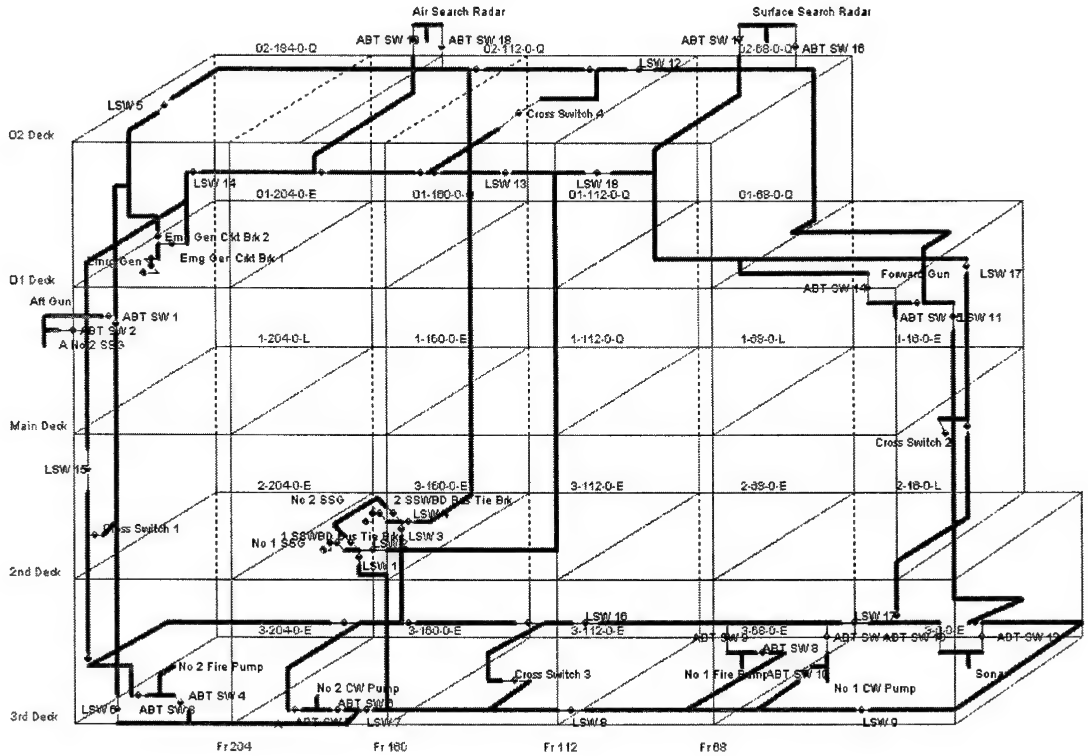


Figure 15. Switched Loop Electrical Distribution Example.

The reliability analysis application was developed by EDD for the analysis of reconfigurable power utility distribution systems. The analysis algorithm first associates components into segments, which are groups of components that are directly affected by any single failure that can be isolated and reconfigured together as a group. It then systematically fails components, and then identifies options for isolation and restoring power. If a switching based reconfiguration option exists for a particular failure, the algorithm uses switching time. For loads that can't be restored through switching, the algorithm uses average repair time for the failed component. Algorithm output includes average downtimes per year for each segment, and an overall averaged down time for the entire circuit.

This analysis provides the capability to associate incremental changes in reliability, cost and efficiency to design choices for adding switches, segments and sources. Current design applications include analysis for optimal placement of distributed generators. Applications under development also include the ability to run reliability for distribution systems that include the use of smart components, which use control concepts similar to the ones used for implementing the use of smart valves on ship. The base analysis features for this algorithm could be modified and extended to perform shipboard interdependent system survivability analysis.

Recoverability Design and Control Analysis

Recoverability: Recoverability, as used in this report, is defined as actions to reconfigure and restore. The concept of Recoverability is central to what makes a system of systems survivable. EDD's LGI based modeling approach provides the base calculation concepts and capabilities

could be used to develop a wide range of different survivability related measures. The following section provides some preliminary definition for two potential reliability based measures.

Recoverability Objective Function – OF_1 Looking for some design that minimizes recovery time and dollars spent

$$OF_1 = \min_j \left\{ \sum_i \sum_k \sum_s \left[LP_s * T_{Rs}(MS_k, e_i, d_j) \right] * b_j \right\} (\sec * \$M)$$

Subject To:

Component equations: $\mathbf{f} = \left(e_i, d_j, \mathbf{V}\right) = 0 \qquad \mathbf{V}(e_i, d_j)$

Where $\mathbf{V}$ = set of through and node variables; these equations depend upon design dj; some of these equations may be removed due to event ei

System Equations:

Node Equations $\mathbf{A}(e_i, d_j, c)\mathbf{V_N} = 0$

Loop Equations $\mathbf{B}(e_i, d_j, c)\mathbf{V_T} = 0$

$\mathbf{V_N}$ = vector of node variables

$\mathbf{V_T}$ = vector of through variables

Inequality Constraints $\mathbf{V_{NL}} \leq \mathbf{V_N} \leq \mathbf{V_{NU}}$

$\mathbf{V_T} \leq \mathbf{V_{TU}}$

Definitions

T_{Rs} – Time to restore s^{th} service due to failures resulting from i^{th} Event e_i, units in seconds

LP_s – Load Priority for s^{th} service that is lost

d_j – j^{th} design

e_i – i^{th} Event (Event is another word for Contingency), specifies failed components

b_j – Cost of j^{th} Design, units in \$ (M - Millions, K – Thousands)

MS_k – k^{th} Mission Set

$C(d_j)$ - Set of configurations of j^{th} Design, $c \in C(d_j)$, c is a set of open – closed, on - off state conditions for switches, loads and sources for some jth design

Load Priority $\mathbf{LP} = \mathbf{z}(MS_k)$ Weighting factor for how important a specific service is to a specific mission set

Cost of jth design $b_j = g(d_j)$

Component Operation Time $\mathbf{To} = \mathbf{h}(d_j)$ **h** is a set of component manual and automatic operation times associate with the components in some j^{th} design

Note: Component equations relate across to through variables. Component equations are specific to an engineering domain. Components can be physical components from multiple engineering domains (electrical, fluid, mechanical, etc,); distributed agent, model based field programmable, or traditional PLC type algorithms that control switching/valve operation and load/source operation/alarm/trip set points; and diagram-able personnel actions. System equations, which are defined by the node and loop equation matrices defined above, are the same across all domains.

Contingency Analysis: Contingency analysis is a well defined survivability type measure used in the Power Utility Industry. Transmission systems are typically designed and operated to be able to sustain any single major component failure or contingency, and still provide services to the rest of the system that is not a direct part of the segment where the failed component resides. A segment is the smallest group of components that can be isolated from the rest of the system by an associated group of switches.

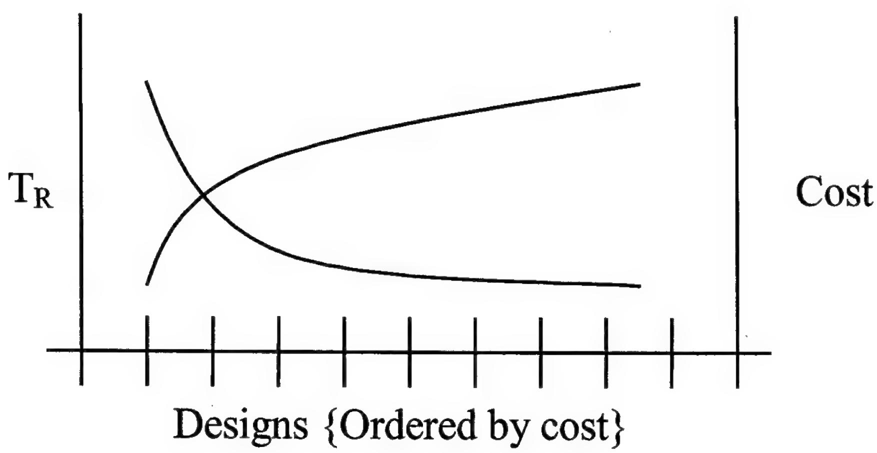
Using EDD's LGI based approach, the number of CoTree elements that any given flow depends on is a direct measure of the service redundancy of the transmission system. This in turn provides a simple measure for Contingency level. A one CoTree dependent flow has level one Contingency. A two CoTree dependent flow has level two Contingency, and so on.

Recoverability Benefit to Cost Ratio

Combining LGI based concepts for assessing Recoverability and Cost provides the capability to automatically generate the following:

$$T_R = \max_{p,s}\left\{\sum_m To_{p,m}\right\}$$

Where $\sum_m To_{p,m}$ = sequenced operations to service s or p^th concurrent restoration activity.



The above graph illustrates that restoration time decreases with an increase in cost. A design should be selected prior to the "flattening" of the restoration time curve.

Recoverability Analysis Software Development Status

The following section provides a step by step narrative of the function and application calls that EDD's LGI model based software would perform for fully automated design and control analysis for an integrated ship system.

Function/Application	Current Software Status
Read in design	The way DEW designates and uses sources greatly simplifies the process required to read in and merge system files. Power Utilities use this feature to merge large system models for both design and on-line control. DEW also has a SOAP interface, which Power Utility customers use to remotely build models as part of collaborative system development and design. It is also used to automatically update system models used for Virtual SCADA applications.
Read in scenario	This includes system alignment, priorities, maneuvering doctrine, minimum capacity, trip and alarm set points. All of the major code pieces for this exist but need to be coordinated together for shipboard application.
Highlight initial failures/damage	Can do this manually. Needs to be automated and linked with scenario read in.
Propagate failures/damage	This is an integral part of current reconfiguration analysis. Display and execution needs to be formatted for shipboard application. Need to create IJC (Interdependency Junction Components) to provide for intersystem analysis.
Manage and set priorities	Currently have a simplified version that manages a set of user specified load priorities. Need to categorize system loads into primary loads whose priorities are assigned directly as a function of specified mission priority, and reconfigurable loads, whose priorities are derived as a function of current system damage and maintenance status, and reconfiguration.
Reconfigure for Restoration	Currently works for individual systems. Need to create IJC's to provide for intersystem analysis. Once IJC's are defined, intersystem analysis would be performed by running reconfiguration and flow analysis for each system separately, when a component in a system is changed because of some related failure or operation change in one system, the IJC propagates the dependency information resulting from that change to the other systems. This process is repeated until reconfiguration and flow for each individual system converges, and there is no new dependency change information remaining to propagate across into other systems.
Evaluate M-Levels for Hit and Recovery	Needs to be created. This would be a simple algorithm that works through sets of lists generated by the failure propagation algorithm.

Automated Reconfiguration and Recovery

Using the functions and applications defined above, the system would perform automated, multiple system reconfiguration and recovery design and control analysis by carrying out the following:

Read in system design, scenario and mission priorities.

Propagate and highlight damage and failures

Evaluate resulting Hit M-Level

Perform multi-system reconfiguration and recovery

Derive priorities for support equipment

Manage trip and alarm settings according to maneuvering doctrine minimal levels

(This looks at total capacity combined system can serve. If capacity drops below a certain set point, the supervisory system resets individual equipment trip points. This could work for field programmable distributed agents as well)

Manage intersystem dependencies

Run reconfiguration for restoration for individual systems

Converge solution for total system using dependency elements

Evaluate resulting M-Level (Rec M-Level)

Determine recovery time (This will be driven by automated switching times combined with personnel routing times to operate manual switches).

DESCRIPTION OF APPROACH

EDD's Linear Graph Iterator (LGI) based modeling and analysis is built around a hybrid combination of concepts from Linear Graph Theory, Physical Network Modeling, and Generic Programming. Linear Graph Theory was founded by Leonhard Euler in the 1700's, and provides a systematic approach for structuring network analysis for any system or process that can be drawn out as a set of arcs (paths) and nodes. Physical Network Modeling was formalized in the 1960's to address interdisciplinary transient and steady state analysis of mechanical, electrical, fluid and thermal systems. Generic Programming is a collection of advanced data management concepts that grew out of the development of the C++ Standard Template Library in the 1990's. Combining these well established engineering concepts with Generic Programming fuses data management and systems analysis together in a way that produces significant breakthrough potential for the advancement of engineering system related design, analysis, management, control, and computation. The most notable feature of the approach is its potential ability to interactively model and analyze physics based component behavior, deterministic control and reconfiguration analysis algorithm driven discrete behavior, and doctrine related manual personnel discrete equipment operation actions across a broad scope of analysis types and engineering problem domains; combined all together into homogeneous interdependent parts of a reconfigurable system of systems. Other benefits include: (1) Computation techniques that eliminate the need to build and manipulate large matrices, (2) Natural, model based analysis driven distributed processing, and (3) Simplified software system development and maintenance.

Together, these features are critical to future understanding and solution of the root problems associated with design, management and control of very large, complex, critical infrastructure type systems. For the Navy, this directly applies to the study and development of survivability

automation, automation for reduced manning, and integrated logistics. A current Army application that fits this area is development and operation of micro-grids. For Homeland Security, critical infrastructure problems include emergency system design and operation for high-rise buildings and subways, and disaster relief resource management. Industry applications include modeling and supervisory control of the national power grid, distributed generation, and combined power and gas transmission and distribution networks, off-shore oil platforms and large industrial facilities.

The Power Utility Industry has been using EDD's LGI based modeling and analysis Distribution Engineering Workstation (DEW) software for distribution system design for ten years. Recent Power Utility use includes: (1) Emergency reconfiguration analysis of a 2000 plus sectionalizing device distribution system to coordinate isolation of a failing critical transformer without violating system overload constraints, and without the loss of any customers, (2) Modeling of a 1.3 million, multiphase component system for which DEW can perform load flow analysis in approximately 45 seconds using a single 2.0 GHz processor, (3) Implementation of a real-time Virtual SCADA installation for monitoring and control of transmission and distribution systems, and (4) Real time control of various types of remote, distributed generators such that system overloads and voltage limit violations are eliminated. The system works by using a detailed model to correlate sparse monitoring data together with detailed historical operational information. Because of the size of the system, and the remote location of the components involved, the "Virtual" system ends up performing better than a traditional SCADA system with extensive numbers of monitoring points and is much cheaper to install and operate. Navy inspection of installed DEW systems can be arranged upon request.

Physical System Modeling

Physical Network Modeling represents components for different systems in terms of their electrical system equivalents, which provides the use of common network analysis methods across different system types. See Tables 1, 2 and 3. It also provides for the application of generic programming. Component variables in physical systems are classified as either across variables or through variables. See Figure 16. .

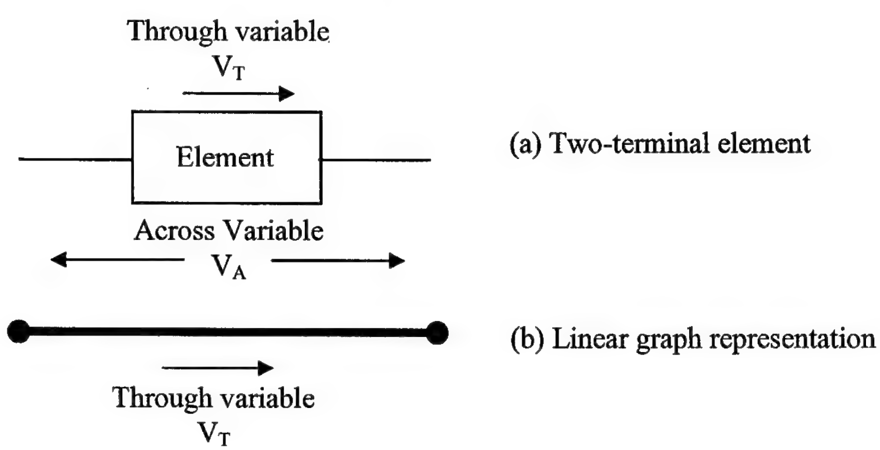


Figure 16. Linear graph representation of two-terminal elements

Across variables include pressure, voltage, and temperature. Through variables include fluid flow, electric current, and heat flow. Across variables are calculated as the difference between the node variables located at terminal ends of a component. Node variables are measured with respect to some common reference point like ground, or atmospheric pressure. Through variables

are measured in reference to flows through the boundaries established by the terminal ends of the components. Systems are modeled by connecting components (or elements) at their nodes. Linear graphs, consisting of elements (or edges) and nodes are used to represent the logical topology of the physical system model. See Figure 18.

System	Through variable V_T	Across variable V_A
Electrical	Current i	Potential difference (voltage) v
Fluid Flow	Volume flow rate Q	Change in pressure P
Mechanical (translational)	Force F	Change in velocity x'
Mechanical (rotational)	Torque T	Change in angular velocity Ω
Thermal	Heat flow rate q	Change in Temperature θ

Table 1. Through and across variables for physical systems

Energy Dissipation Elements

System	Constant	Element equation
Electrical	Electrical Resistance R	$v = \mathrm{R}\, i$
Fluid Flow	Fluid Resistance $\mathrm{R_f}$	$P = \mathrm{R_f}\, Q$
Mechanical (translational)	Translational Damping Coefficient B	$x' = F / \mathrm{B}$
Mechanical (rotational)	Rotational Damping Coefficient $\mathrm{B_r}$	$\Omega = T / \mathrm{B_r}$
Thermal	Thermal Resistance $\mathrm{R_t}$	$\theta = \mathrm{R_t}\, q$

Table 2. Energy dissipaters.

Component Equations: Across and through variables for individual elements are related together by system type specific component equations: V_A = Constant * V_T

For example, for electrical systems v (across variable) = R * i (through variable). More complicated effects including nonlinearities can be included in these equations and do not affect convergence [1].

System Equations: System equations for all system element types are the same, and obey the following requirements:

- The sum of the across variables around a loop equals zero
- The sum of through variables at a node equals zero.

This commonality provides the theoretical base for saying it is possible to write a common set of generic algorithms that can operate on systems belonging to different engineering domains.

Energy Storage Elements

Energy Storage elements obey the following equation: $y = k\frac{dx}{dt}$

There are two types of energy storage elements, which depend on how energy is stored.

I. A-type energy storage (storage through the across variable)

$$V_T = \text{Constant} * (\frac{d}{dt}V_A)$$

II. T-type energy storage (storage through the through variable)

$$V_A = \text{Constant} * (\frac{d}{dt}V_T)$$

System	A-type energy storage		T-type energy storage	
	Constant	Element equation	Constant	Element equation
Electrical	Electrical Capacitance C	$i = C * (\frac{d}{dt} v)$	Inductance L	$v = L * (\frac{d}{dt} i)$
Fluid Flow	Fluid Capacitance C_f	$q = C_f * (\frac{d}{dt} P)$	Fluid Inertance I	$P = I * (\frac{d}{dt} Q)$
Mechanical (translational)	Translational Mass m	$F = m * (\frac{d}{dt} x')$	Translational Spring Coefficient k	$x' = \frac{1}{k} * (\frac{d}{dt} F)$
Mechanical (rotational)	Inertia J	$T = J * (\frac{d}{dt} \Omega)$	Rotational Spring Coefficient k_r	$\Omega = \frac{1}{k_r} * (\frac{d}{dt} T)$
Thermal	Thermal Capacitance C_t	$q = C_t * (\frac{d}{dt} \theta)$		

Table 3. A-type and T-type energy storage (Thermal system has no equivalent of inductance)

The following transient equations apply to energy storage equations: $y = k\frac{dx}{dt}$

Numerical methods, including the Euler and Runge-Kutta methods, can be applied to solve for x and y.

$$y = k\frac{dx}{dt} \quad \text{with initial condition of } y_o \text{ when } x = x_o$$

at $t = \Delta t$

$\Delta x_1 = (y_o / k) * \Delta t$ then $x_1 = x_o + \Delta x_1$

y may have a new value y_1 (depending on surroundings, needs to be iterated)

at $t = 2\,\Delta t$

$\Delta x_2 = (y_1 / k) * \Delta t$ then $x_2 = x_1 + \Delta x_2$

Given a set of independent across and/or through variables for the system, all dependent variable values (across and through) can be written in terms of the given set of independent variables. However, there are many different sets that can be used for the set of independent variables. In a linear graph where a tree has been selected, independent variable sets can be defined using either the node variables or the cotree through variables. That is, if a set of measurements of the tree node variable values are made, then all other node and through variable values can be calculated using the system equations. Or, if a set of measurements of the cotree through variables are made, then all other node and through variable values can be calculated. Thus, the tree nodes provide a set of independent node variables and the cotree elements provide a set of independent through variables.

Based on the concepts listed above, models of physical systems across engineering domains have the following in common:

- Linear graphs may be used to model the arrangement and connection of components
- Node or across variables
- Through variables
- Given any tree in a linear graph, the across variables for the tree form an independent set of variables
- Given any cotree for a linear graph, the through variables for the cotree form an independent set of variables

The application of generic programming to physical systems builds on these commonalities.

Generic Programming

Generic programming is built around the use of algorithms, containers, and iterators. Generic algorithms provide the capability to reuse code across multiple domains. For example, the sorting algorithm in the C++ Standard Template Library (STL) can be used to sort any Type of objects as long as the Type provides greater than/less than operators. To apply an algorithm to a new domain, all a programmer has to do is define new Type operators. Generic Programming was originally developed to improve information systems data management. Use of the Physical Network Modeling principles described in the previous section makes it possible to apply Generic Programming concepts to engineering system design and control. Based on EDD's current work with the power utility industry and a literature search, the LGI based approach used in EDD's software is the only know research and/or commercial software application that combine Physical Network Modeling and Generic Programming together into a unified critical infrastructure systems design and control analysis approach. Based on current Power Utility results, along with recent Gas Utility and Computing Industry interest, further work to formalize the combination of these two approaches holds tremendous potential for radically changing the way complex system research, design and control are done. The following section describes the combined generic programming physical network concepts EDD has developed.

Containers: Figure 17 illustrates an aggregation relationship (indicated by open diamond on the container) between a Container and Some Class. Examples of containers include stacks and vectors. Containers are generic storage devices for large sets of objects. The use of a container greatly simplifies data management for complex processes that interactively store and use large amounts of data across multiple domains. For the critical infrastructure problem, objects representing the individual components of a collection of interdependent systems, like electrical distribution, piping, ventilation, compartmentation, etc. for a naval ship; are stored together as homogonous system of systems elements in a "Network" container.

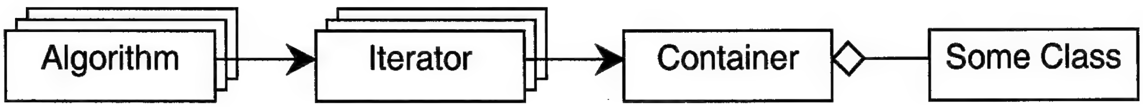


Figure 17. Generic Programming Paradigm

References and Iterators: Prior to discussing iterators, the concept of local references needs to be understood. Local references are used to build iterators when components are stored in the Network.

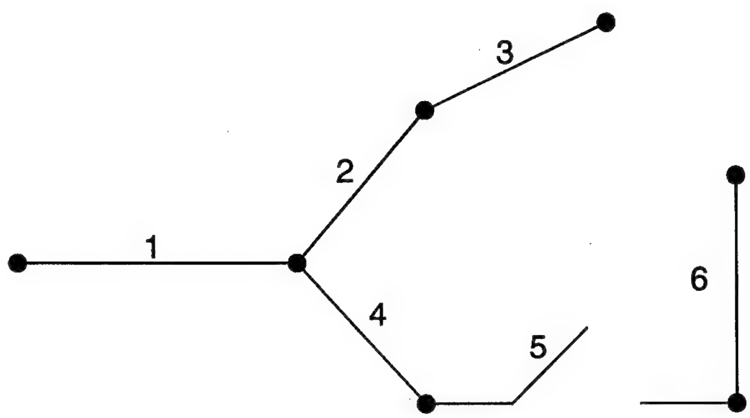


Figure 18. An element-element linear graph with unique identifiers on each element in the graph

Consider the linear graph shown in Figure 18. There are six elements in the graph, numbered 1 through 6. Assume element 1 corresponds to a source element. Also assume that Element 1 is the only source in the graph and as such serves as the reference source for all elements in the graph. Element 1 maintains a set of references to the elements that it physically connects to. At its ending node Element 1 knows that it connects to elements 2 and 4. At its starting node, element 1 connects to a reference node. Likewise, Element 2 knows that it connects to elements 1 and 4 at its starting node (which is the point closest to its reference source) and to Element 3 at its ending node.

In the element-element implementation of a linear graph, each element carries with it a unique reference source and an ending node. The ending node of the element is the point on the element that is the most distant from its reference source. Thus, the measurement of a node variable for an element occurs at the point that is most distant from its reference source, and the node variable for an element is uniquely defined. Thus, the Network container only needs to manage one type

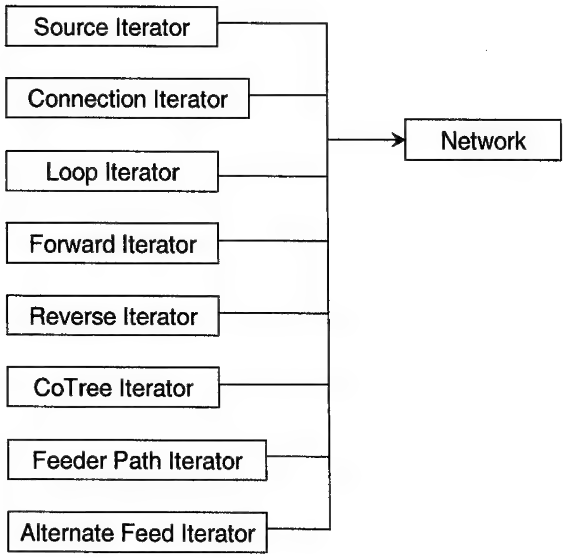


Figure 19. Types of Iterators

of object, elements, instead of two, elements and nodes. References are the base elements used to define Iterators.

Each iterator type shown in Figure 19 corresponds to some type of list of objects. Algorithms use iterators to retrieve and process objects in a container. Iterators are pointers or pointer-like objects that can be used to process objects stored in a container. The linear graph of the physical system model, which can be automatically generated directly from an object based drawing of the system, can then be used to automatically construct and manage iterators. This means that updating the drawing, for both design changes and reconfiguration switching, in essence also fully updates the iterators which drive the integrated systems analysis, control simulation and dynamic data management used to analyze, monitor and control the system. These concepts will be explained in more detail through the remainder of this section.

Figure 19 illustrates the relationship between various types of iterators and a Network container storing components (where component means the same as element, but the word component is used when the element represents a component in a physical network). For notational simplicity, parameterized class notation on the iterators has been dropped.

Various types of iterators may be used to process objects of type Component stored in a Network container. Each iterator shown in Figure 19 may be used to return a particular set of objects, one-by-one. The Source Iterator shown in Figure 19 may be used to process through a list of all sources in the Network. For a given source, the Alternate Feed Iterator may be used to process alternative feeds for the given source. The Cotree Iterator may be used to process a list of cotree components for the linear graph. That is, the Cotree iterator returns components, one-by-one, such that if all their through variables are known, then all through variables in the graph may be calculated. Some of the iterators shown in Figure 19 must be initialized first by accessing other iterators. For instance, the Forward Iterator works from a given source component. To initialize a Forward Iterator, a source component must be provided. The Forward Iterator may then be used to access all components for which the given source is the reference source. For a given source, the Reverse Iterator processes the components in the exact opposite order of the Forward Iterator. The first component returned by the Forward Iterator is always physically connected to the source. However, a component returned by the Forward Iterator does not have to be physically connected to the previous component returned by the Forward Iterator.

Network Object Concepts: Figure 20 is a UML presentation of a Physical Network Model – Generic Programming software framework. Here the Network container is thought of as storing the elements of a linear graph, and the iterators provide a means for traversing through those elements. In order to implement this, the linear graph is considered to consist of just elements. The linear graph concept introduced in previous sections worked in terms of both elements and nodes. This may be referred to as an element-node graph implementation. Another type of graph implementation is an element-element implementation.

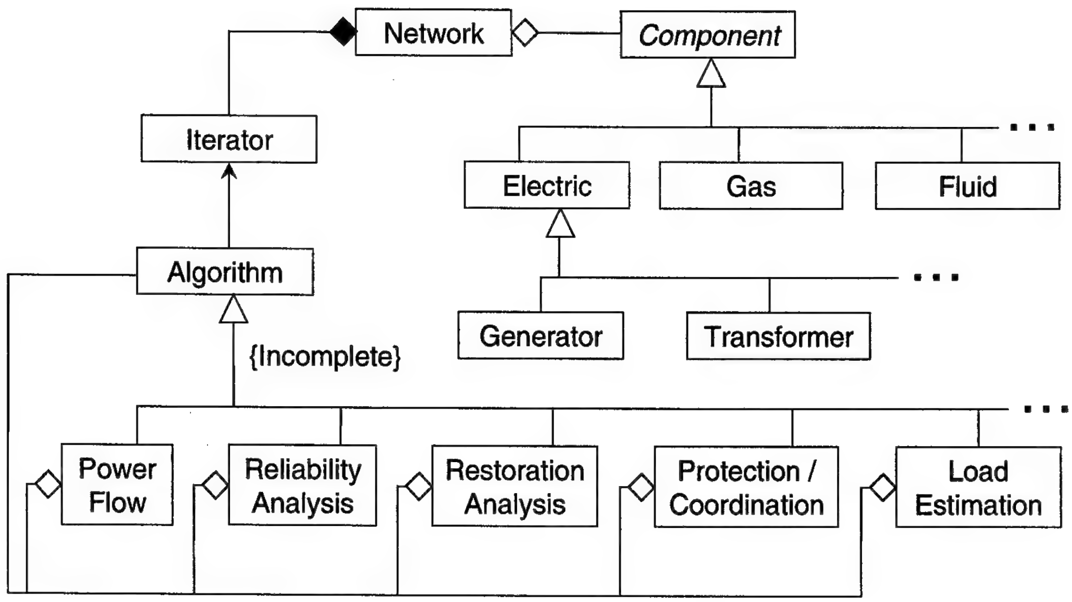


Figure 20. Physical Network Modeling – Generic Programming Unified Systems Framework

The parameterized algorithm class shown in Figure 20 serves as a super class for a set of algorithms that operate on a particular class of objects, such as Component, that can be Electric, or Gas objects. That is, some algorithms, such as restoration analysis, could simultaneously process both electric and gas type objects. Furthermore, this allows for sharing of common algorithms among different types of objects. Examples of such common algorithms include across variables (such as voltage, pressure, or temperature differences) around a loop summing to zero or through variables (such as electric current, gas flow, fluid flow, or heat flow) at a connection (node with zero storage) summing to zero.

A fundamental concept of object oriented analysis is that the reason for existence of all objects should be identified, and that all behavior and data contained in the object should support the reason for its existence. This concept can be applied to the solution of physical networks, where the network itself is considered to be an object. Previous works have applied object oriented analysis to the solution of physical networks, but in general they only treated the components that make up the network as objects. The entire network itself has not previously been considered to be an object. This concept greatly simplifies integration across different engineering domains and reduces the need for matrix based system equations.

Let us first consider responsibilities for an object which models a physical component in a network. The object should have the responsibility of providing a consistent set of terminal conditions for itself (ie, variables that the external world can measure about the object by connecting a measuring instrument to its terminals). That is, the object should be able to calculate its across variables given its through variables, or calculate its through variables given its across variables, or some hybrid combination thereof. The object may have different states, such as open or closed or failed, and should provide the appropriate response based upon its state.

Responsibilities of a network object would be to ensure that all through variables at each node in a set of independent nodes sum to zero and that all across variables around each loop in a set of independent loops sum to zero. From this perspective, it is possible to have a single network type represent networks across engineering disciplines. That is, a network object can represent an electrical network and a gas network together.

A network object is actually a container for component objects. For instance, a network object representing an electrical power system would contain power source, line, transformer, and capacitor objects. These objects stored in the network container can be accessed via different types of iterators [2, 3]. Algorithms, such as power flow, fault, or reliability analysis, use the iterators to access the objects stored in the network container.

Generic programming uses the concept that objects are placed in containers and that algorithms use iterators to access the objects. A network class for a particular engineering discipline can be implemented with a class template. Being able to use the same network class to manage different engineering problem domains provides the foundation needed for reconfiguration across interdependent systems representing different domains [7].

Iterator Based Analysis: System analysis algorithms that are written in terms of iterators represent a departure from the main stream approach that relies upon matrices. However, iterators provide a number of advantages over the matrix based approach, including the following:

- Iterators are better suited for handling design changes to a system
- Iterators are better suited for handling configuration changes to the system (for instance, switches being opened or closed)
- Iterators may be used to dynamically maintain independent sets of system equations, such as an independent set of loop equations
- Distributed computation can be naturally implemented with iterators [4]
- Since a common model (stored in the network container) is used by all analysis algorithms, a unified approach to system analysis is provided
- A collaborative design environment is naturally supported

Each of these concepts will be considered in turn. Engineering design involves evaluating and comparing alternatives defined in terms of performance, reliability, safety, efficiency, cost, and others. With iterators, it is easy to insert, delete, and add component objects into a system (ie, a network container), and then evaluate the affects of that change. When a physical change is made to the system, only the local iterators related to that change need to be updated. Furthermore, this approach provides a natural environment for design algorithms that seek optimum solutions [5]. When the configuration of a system changes, such as a switch opening, again only the local iterators need to be updated. This eliminates the need for special processing for design and system reconfiguration analysis. With a matrix approach, when the configuration changes, the matrices have to be reformulated. For a large system consisting of millions of objects, the matrix approach is faced with significant computation just to process the configuration change.

Iterators may be used to dynamically maintain independent sets of node and loop equations associated with switching and design change. For instance, when a switch is closed in an electrical system, a loop may or may not be created. However, when the switch is closed, two feeder path traces that run from either side of the switch closure may be used to identify if a loop has been created [6]. If the feeder path traces either intersect or trace back to sources, then a loop has been created. When a loop is created, the switch is marked as a cotree element, and the cotree element is added to a set of cotree iterators. Note that a cotree element may be created whenever two components are connected together. Thus, a cotree element could represent any type of device and not just a sectionalizing device.

Computation Speed: Most physical systems have many more independent node variables than independent loop variables. Matrix based approaches commonly solve for node variables. An Iterator approach provides the capability to solve for cotree loop flows, which greatly reduces problem size and increases computation speed. The use of Iterators also provides a natural, network problem driven process for structuring distributed processing, which also increases

computation speed [4]. Natural points to split the computations occur at cotree elements. In addition, with the iterator approach only variables at the boundaries where the network is divided between processors need to be exchanged. Hence, there is very little communication overhead. With the matrix based approached, methods for distributing the computations requires specialized machines.

Other Benefits: Having all analysis based upon a common model and the same iterators provides a unifying approach to system analysis. With matrix based approaches, special models are developed for each type of analysis to be performed, such as power flow, fault, or reliability analysis. In the matrix approach, the model is embedded in the algorithm that solves the problem. This adds complexity and makes software maintenance difficult as each special type of analysis must be updated when new types of components are added to the analysis. With the generic approach, a new component object can be added and none of the algorithms operating on the model (power flow, fault, or reliability) need to be updated.

Generic programming provides for different algorithms being attached simultaneously to the network container. Hence, two different design calculations written by different engineers could play together on the same network. This forms the basis for a collaborative design environment that provides many opportunities [5].

The generic programming approach has proven itself in the field on large system problems [8]. It has been used in actual real-time operations to solve reconfiguration problems involving more than 2000 sectionalizing devices. It has also been used to solve power flow problems involving over 1.3 million, multiphase components with a solution time on a single processor of approximately 45 seconds. Using this approach, there is no theoretical limit to the size system that may be solved.

INDUSTRY USE OF LGI BASED MODELING

Virtual SCADA System: EDD performed this work as a subcontractor under Detroit Edison for DOE. The project involved development and implementation of a real-time "Virtual SCADA" installation for monitoring and control of transmission and distribution systems with remote distributed generators (DGs). The system works by using a detailed model to correlate sparse monitoring information together with detailed historical customer load data. Because of the size of the system and the remote location of the components involved, the Virtual system provided better performance than a traditional SCADA system with extensive numbers of monitoring points, and was much cheaper to install and maintain.

The system is currently run as a monitoring and decision aid for use by a system operator. A real-time database collects measurements at the start of circuit and at the DG installation sites, and provides them to the control software. The control software houses an accurate model of the circuit that uses both on-line measurements and extensive historical load data. The model based analysis engine built into the control software uses the circuit data and real-time measurements to predict flows and check for likely overload and under voltage problems for all of the modeled areas under current conditions. These predictions have proven to be very accurate and provide actionable data for areas that are difficult to directly monitor. When a problem is detected, the analysis engine simulates possible circuit flows that would occur if DG generation were varied. This is done to determine the DG generation level that will best correct the problems. The calculated generation level is then sent back to the real-time database as a recommendation for KW and KVAR set points for operating the DG. The system operator can then implement the recommendations by communicating the set points to the local controllers installed with the DG in the field.

The next planned development phase for this project is to use the approach to aggregate multiple DGs installed in various locations throughout a system, into a coordinated group of generation under Level 2 control that could then be made available to transmission system control. When a desired generation set point is provided to the Level 2 DG controller, it would then calculate the most economic way to provide that generation with the DG generation units at its disposal. The cost of generation will be planned by the aggregator. The same type of Virtual SCADA monitoring and control system could be used to coordinate efficiency and survivability configuration and set point control for shipboard systems.

Million Node Model: The largest system modeled to date with DEW is a 1,300,000 node, 100,000 sectionalizer device, 4000 square mile urban section of Detroit. DEW can perform power flow for this system in 45 seconds using a single 2.0 GHz processor. See Figure 21. The DEW display shown in Figure 21 was developed specifically for distribution system design. The reference model based techniques used to implement it greatly simplify display and user interface modification. Utility companies often use their own custom symbols and highlighting schemes. Specific component details can be viewed by zooming in on any selected area. Failed sections, restoration paths, and selected performance parameters can also be highlighted and displayed. Models can be drawn directly in DEW or downloaded from other software systems. This particular model was created from a CAD drawing generated at Detroit Edison. DEW was recently used to reconfigure a large distribution section in St Louis, Missouri, to isolate a failing substation transformer without any customer power outages, and is currently being installed by Con Edison in New York to perform supervisory control for the Orange and Rockland distribution system. Department of Energy sponsored projects are also underway to implement real-time reference model based control, monitoring and reconfiguration analysis. Current utility work also includes the development of reference model based integrated monitoring and control of distributed generation, which has many similarities with shipboard power systems.

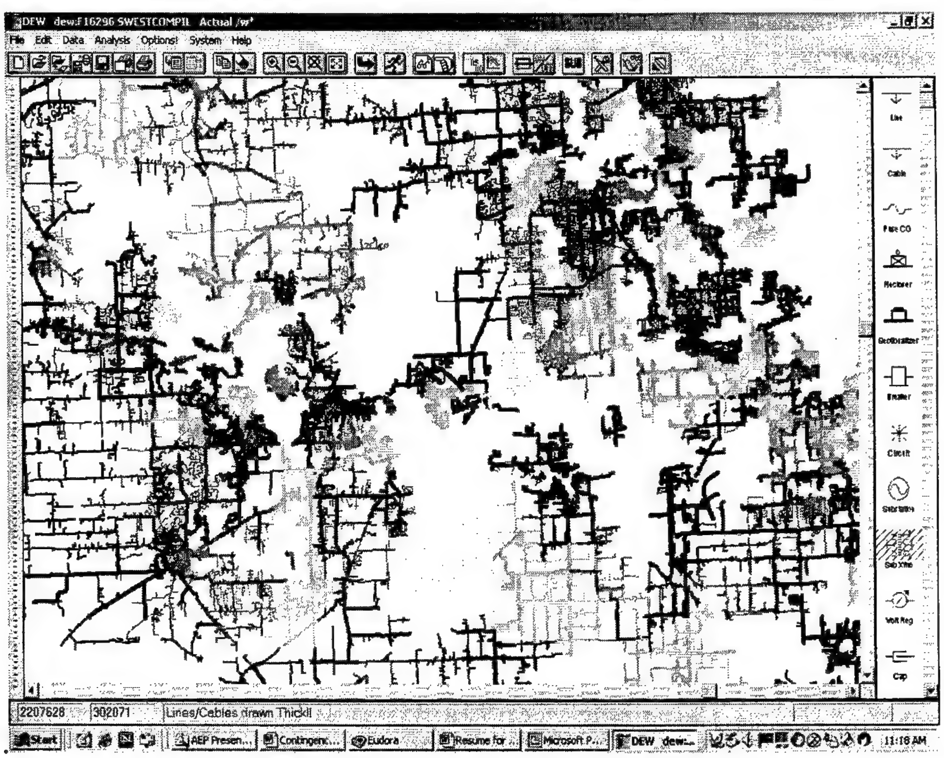


Figure 21. 1,300,000 Node Power Utility Distribution System

Fault Analysis

A power utility recently collected fault data for a distribution system they already had modeled in DEW. The following analysis compares measured fault data to DEW model predictions.

Fault Data: Recorded Fault Currents at Clark Circuit

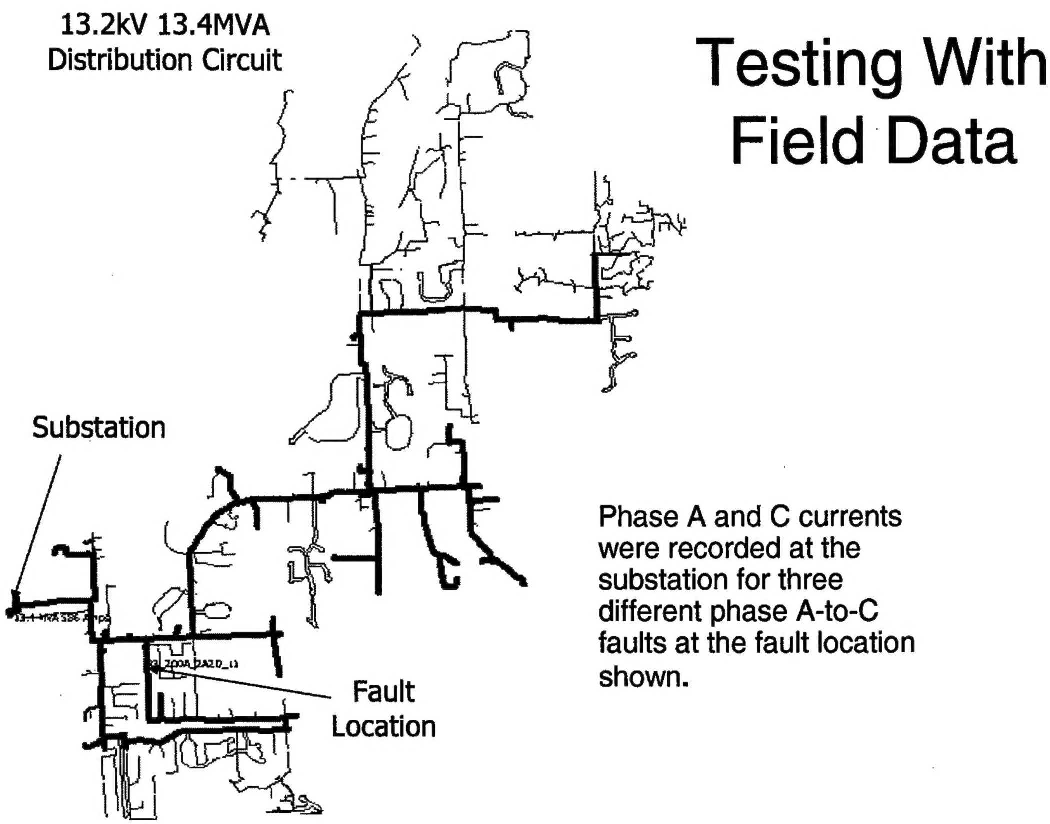
Fault Location => recloser - local name: R3_200A_2A2D_L1 (x=2249505 ft, y=452820 ft)

Recorded Fault Currents at the start of the circuit for Phase-A-to-C Fault at the recloser:

First Fault	Ima = 2731 Amp, Imc = 3114 Amps.
Second Fault	Ima = 2738 Amp, Imc = 3111 Amps.
Third Fault	Ima = 2680 Amp, Imc = 2999 Amps.

Note: These recorded currents include both the fault current and load currents.

Clark Circuit Model:



[redacted]

TECHNOLOGY DEVELOPMENT AND TRANSFER

EDD has been providing linear graph iterator based modeling and analysis consulting and software for Utility Industry collaborative distribution and transmission system design, analysis and control research and development for ten years. EDD routinely works with a diverse collection of system design and control engineers, university students and professors, and small and large technology companies; providing core systems analysis and primary software architecture for collaborative applied systems research and control. EDD is currently providing this type of service for several Power Utility and DOE funded projects.

The generic nature of EDD's approach, coupled with it focus on detailed component modeling, makes it very extensible. This has resulted in a large and versatile collection of highly integrated, highly interchangeable analysis applications, concepts and data that cuts across several major areas of Power Utility design, operation and control. The advanced systems analysis development needed by the Navy to automate survivability and system integration for reduced manning fits well with recent Power Utility, Homeland Security, Department of Energy, and Computing Industry interest in modeling and control of very large integrated systems. EDD's LGI based approach has already been accepted for use by several Industry leaders in this area and is under consideration for use by several more.

These areas include:

- National power grid modeling and supervisory control
- Distributed Generator system design, monitoring and control
- Advanced distributed processing and simulation
- Virtual SCADA
- Model based field programmable agents
- Combined Gas and Power Utility system design, monitoring and control
- Analysis engine development for major market software

Many of the concepts being developed for these areas are directly applicable to integrated ship system design and control. Navy use of EDD's LGI based approach would provide significant synergy potential for concurrent systems analysis and control development with Industry.

Specific Navy Program and Industry visits conducted during this contract included discussions with Northrop Grumman's Carrier Technologies Group at VASCIC (The Virginia Advanced Shipbuilding and Carrier Integration Center), and the Naval Surface Warfare Centers (NSWC) Carderock AMMARS (Automated Maintenance Management and Assessment Readiness System) Innovation Group. EDD and Northrop Grumman identified several potential aircraft carrier and destroyer electrical system applications that are not currently being addressed, that would work well with EDD's LGI model based approach. NSWC's AMMARS group identified several advanced concepts critical to future integrated logistics system development that LGI based modeling could be used to develop. These include model based architectures, model based reasoning, and high level reconfiguration analysis.

CONCLUSION

The work performed under this contract demonstrated significant potential for application and further development of EDD's Linear Graph Iterator (LGI) model based analysis approach for integrated shipboard system reconfiguration and recovery design and control analysis. Primary accomplishments competed for this contract include development of two example shipboard system models to demonstrate basic LGI model based analysis concepts, and a significant amount of recovery analysis problem definition and integrated systems analysis software architecture definition work.

EDD's LGI based modeling and analysis approach combines well established Linear Graph and Physical Network multidiscipline analysis approaches with state of the art generic programming principles that have significant potential for producing breakthrough improvement in simulation based analysis for large critical infrastructure type systems. As EDD and its growing network of industry partners continue to work in this area, it is reasonable to assume that over the next three to five years they will radically change the way complex critical infrastructure system design and control is done. Navy participation in development of this approach could greatly benefit both the Navy and U.S. Industry, as they both work to realize complex systems that are both efficient and survivable.

Traditional Navy ship design, control and management are primarily built around survivability principles. Design, control and management for commercial systems focus on efficiency, and are driven by profit motive. The traditional core design principles for survivability and efficiency are mutually exclusive. Past efforts to design a system that compromises between the two states by mixing traditional survivability and efficiency principles together generally produces a system that is neither survivable nor efficient. Advances in technology and the high long-term cost of personnel have driven the Navy to look at integrating advanced systems across areas that used to be designed and implemented separately. On the Industry side, terrorism and the desire to be able to survive natural disasters and other major problems is driving commercial systems to be more survivable. Rising personnel costs have a similar affect in industry because increasing survivability using traditional systems typically requires a lot of well trained, highly experienced people. In order to build a system that is both survivable and efficient, it needs to be formulated as a complex system problem centered around understanding and manipulating inter-system dependencies. The use of traditional analysis approaches, which divide things up along existing discipline and process specific lines, breaks up and obscures the complex interactive behavior that you need to be able to understand and control. EDD's Linear Graph Iterator model based approach provides the capability to directly simulate, analyze and control complex interdependent components (here the word component includes hardware, software and discrete packets of personnel related behavior) together in one common domain. The common model produced by this approach becomes the primary source for optimized analysis and control information for the entire system, which is a capability not currently available from other approaches.

REFERENCES

1. R. Broadwater, "A Design Approach for a Power Plant Feedwater Control System," *IEEE Control Systems Magazine*, 3(1), 1983, 2-11
2. M. Dilek, R. Broadwater, "An Introduction to Integrated Design in Electrical Distribution," *Proceedings of PES 2002 Winter Meeting*, January 27-31, 2002, New York, NY.
3. F. Li, R. Broadwater, "Distributed Algorithms with Theoretic Scalability Analysis of Radial and Looped Load Flows for Power Distribution," *Electric Power Systems Research*, volume 65, issue 2, pp. 169-177, 2003.

4. F. Li, R. Broadwater, "Distributed Algorithms with Theoretic Scalability Analysis of Radial and Looped Load Flows For Power Distribution," *Electric Power Systems Research*, volume 65, issue 2, pp 169-177, March 2003.
5. F. Li, R. Broadwater, "Software Framework Concepts for Power Distribution System Analysis," *IEEE Transactions on Power Systems*, Vol. 19, No. 2, pp 948-956, May 2004.
6. R. P. Broadwater, J. C. Thompson, T. E. McDermott, Pointers And Linked List In Electric Power Distribution Circuit Analysis, *Proceedings of 1991 IEEE PICA Conference*, pp. 16-21, Baltimore, Maryland, May 7-10, 1991.
7. T. E. McDermott, R. Broadwater, "A Heuristic Nonlinear Constructive Method for Distribution System Reconfiguration," *IEEE Transactions on Power Systems*, Vol. 14, No. 2, pp. 478-483, May 1999.
8. A. Sugg, R. Seguin, and C. Scirbona, "Three Utilities Implement Packaged Distribution System," *Transmission & Distribution World*, September 2002, pp. 52-58.